

Phase 5: Performance Testing

DATE	1 NOVEMBER 2025
TEAM ID	NM2025TMID06105
PROJECT NAME	Calculating Family Expenses using ServiceNow

1. Testing Objective:

The objective of testing is to ensure that the Family Expense Tracker in ServiceNow works correctly, calculates totals accurately, and updates records in real-time without errors.

2. Types of Testing Performed:

Testing Type	Description
Unit Testing	Tested each table and field individually
Functional Testing	Verified all features work as expected
Integration Testing	Checked link between Daily Expense & Family Expense tables
Validation Testing	Ensured mandatory fields & correct data entry
Performance Testing	Verified system updated totals instantly
User Acceptance Testing (UAT)	Ensured the solution is user-friendly & meets objectives

3. Test Cases:

Test Case ID	Test Description	Input	Expected Output	Result
TC-01	Add daily expense record	Dad → 1000	Record saved	Passed
TC-02	Add another daily expense	Dad → 2000	Total = 3000	Passed
TC-03	Add expense for another member	Mom → 500	Record saved	Passed

TC-04	Check total for Mom	Mom → 500	Correct total	Passed
-------	---------------------	-----------	---------------	--------

TC-05	Check auto-generated number	Save record	Auto-ID generated	Passed
TC-06	Invalid amount entry	Blank amount	Error shown	Passed
TC-07	Submit without date	Blank date	Error shown	Passed

4. Performance Metrics:

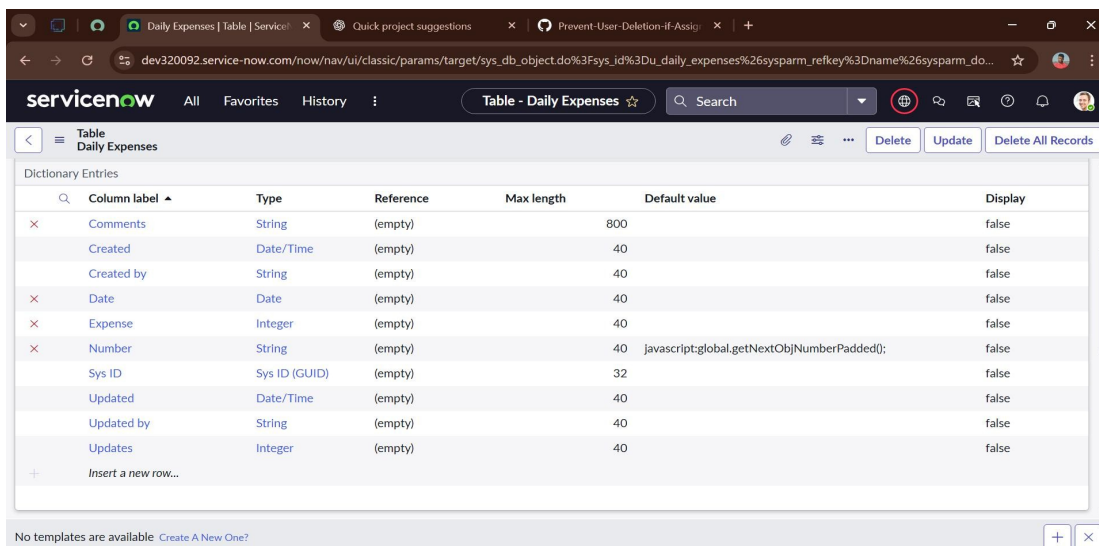
Parameter	Result
Record Save Time	Less than a second
Total Calculation Time	Instant update
Page Load Speed	Smooth
System Load Handling	Stable for multiple entries

5. Observations & Findings:

- Daily expenses correctly update Family Total
- Business Rule triggers worked properly
- Auto-Numbering works
- Reference field links correctly
- No performance lag
- Data remains consistent after multiple entries

6. Pictures for references:

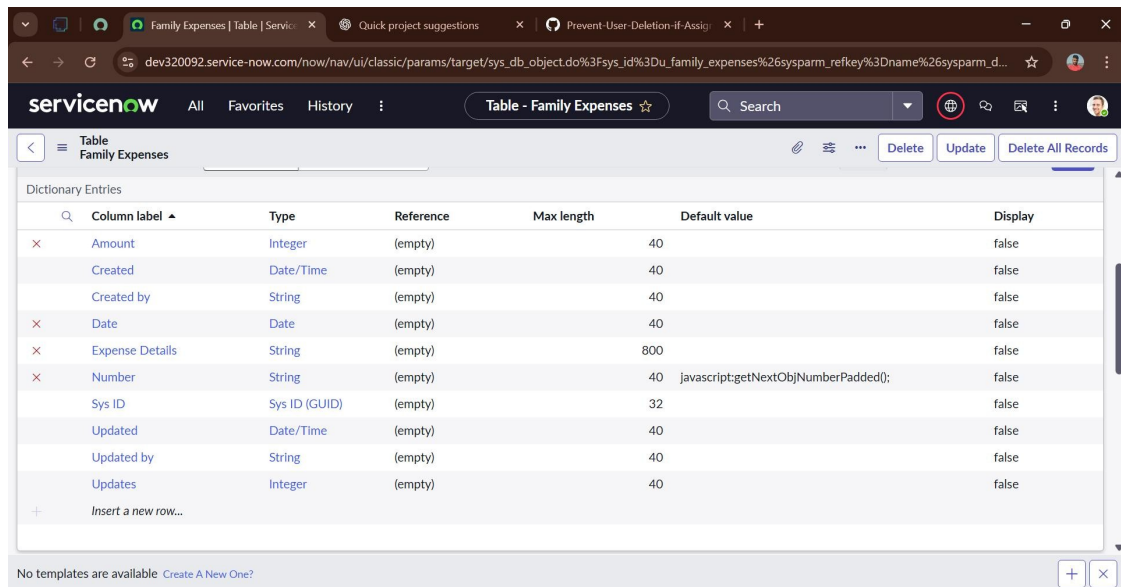
6.1. Daily Expense Table:



The screenshot shows the ServiceNow interface for the 'Table - Daily Expenses'. The 'Dictionary Entries' table is displayed with the following columns: Column label, Type, Reference, Max length, Default value, and Display. The table contains 13 rows of data, including fields like Comments, Created, Created by, Date, Expense, Number, Sys ID, Updated, Updated by, and Updates. Each row has a corresponding 'X' icon in the first column, likely for deleting the entry. The 'Default value' for the 'Number' field is a JavaScript function: `javascript:global.getNextObjNumberPadded();`.

Column label	Type	Reference	Max length	Default value	Display
Comments	String	(empty)	800		false
Created	Date/Time	(empty)	40		false
Created by	String	(empty)	40		false
Date	Date	(empty)	40		false
Expense	Integer	(empty)	40		false
Number	String	(empty)	40	javascript:global.getNextObjNumberPadded();	false
Sys ID	Sys ID (GUID)	(empty)	32		false
Updated	Date/Time	(empty)	40		false
Updated by	String	(empty)	40		false
Updates	Integer	(empty)	40		false

6.2. Family Expense Table:

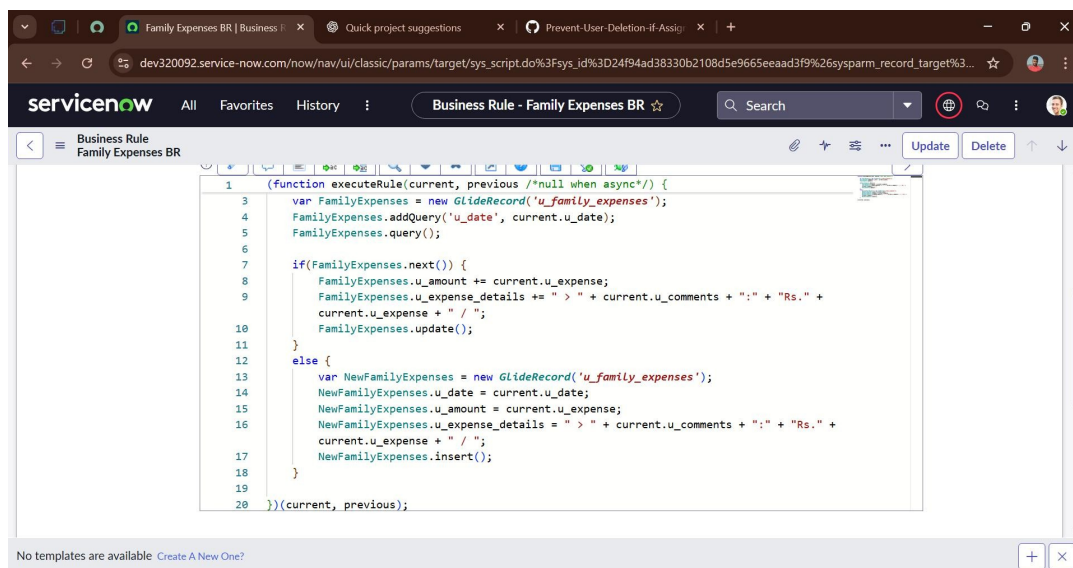


The screenshot shows the ServiceNow interface for configuring a table named 'Family Expenses'. The 'Dictionary Entries' section is expanded, displaying a list of fields with their properties.

Column label	Type	Reference	Max length	Default value	Display
Amount	Integer	(empty)	40		false
Created	Date/Time	(empty)	40		false
Created by	String	(empty)	40		false
Date	Date	(empty)	40		false
Expense Details	String	(empty)	800		false
Number	String	(empty)	40	javascript:getNextObjNumberPadded();	false
Sys ID	Sys ID (GUID)	(empty)	32		false
Updated	Date/Time	(empty)	40		false
Updated by	String	(empty)	40		false
Updates	Integer	(empty)	40		false

At the bottom of the table, there is a link to 'Insert a new row...'.

6.3. Business Rules Script:



The screenshot shows the ServiceNow interface for a Business Rule script named 'Family Expenses BR'. The script is written in JavaScript and is designed to update or insert records into the 'Family Expenses' table based on the current record's 'u_date'.

```
1 (function executeRule(current, previous /*null when async*/) {
2
3   var FamilyExpenses = new GlideRecord('u_family_expenses');
4   FamilyExpenses.addQuery('u_date', current.u_date);
5   FamilyExpenses.query();
6
7   if(FamilyExpenses.next()) {
8     FamilyExpenses.u_amount += current.u_expense;
9     FamilyExpenses.u_expense_details += " > " + current.u_comments + ":" + "Rs." +
10    current.u_expense + " / ";
11    FamilyExpenses.update();
12  }
13  else {
14    var NewFamilyExpenses = new GlideRecord('u_family_expenses');
15    NewFamilyExpenses.u_date = current.u_date;
16    NewFamilyExpenses.u_amount = current.u_expense;
17    NewFamilyExpenses.u_expense_details = " > " + current.u_comments + ":" + "Rs." +
18    current.u_expense + " / ";
19    NewFamilyExpenses.insert();
20  }
21 })(current, previous);
```

6.4. Input in Daily Expenses

Daily Expenses | ServiceNowQuick project suggestionsPrevent-User-Deletion-If-Assig

dev320092.service-now.com/now/nav/ui/classic/params/target/u_daily_expenses_list.do

servicenowAllFavoritesHistoryAdminDaily ExpensesSearch

Daily ExpensesNumberSearchActions on selected rows...New

All

	Number	Date	Expense	Comments
<input type="checkbox"/>	DEF0001019	2025-10-30		1,000 Dad
<input type="checkbox"/>	DEF0001023	2025-10-29		100
<input type="checkbox"/>	DEF0001022	2025-10-31		500 Mom
<input type="checkbox"/>	DEF0001018	2025-10-30		2,000 Dad

<<<1 to 4 of 4>>>

6.5. Output in Family Expenses:

Family Expenses | ServiceNowQuick project suggestionsPrevent-User-Deletion-If-Assig

dev320092.service-now.com/now/nav/ui/classic/params/target/u_family_expenses_list.do

servicenowAllFavoritesHistoryAdminFamily ExpensesSearch

Family ExpensesAmountSearchActions on selected rows...New

All

	Number	Amount	Date	Expense Details
<input type="checkbox"/>	MFE1015	100	2025-10-29	> :Rs.100 /
<input type="checkbox"/>	MFE1014	500	2025-10-31	> Mom:Rs.500 /
<input type="checkbox"/>	MFE1013	2,000	2025-10-30	> Dad:Rs.1000 / > Dad:Rs.2000 /

<<<1 to 3 of 3>>>